The Earth System Modeling Framework

Arlindo da Silva*, C. DeLuca†, V. Balaji‡, C. Hill§, J. Anderson‡,
B. Boville†, N. Collins†, T. Craig†, C. Cruz*, D. Flanigan†, B. Hallberg‡,
M. Iredell¶, R. Jacob‖, P. Jones**, B. Kauffman†, E. Kluzek†, J. Larson‖,
J. Michalakes†, D. Neckels†, W. Sawyer*, E. Schwab†, S. Smithline‡, Q. Stout††,
M. Suarez*, A. Trayanov*, S. Vasquez†, J. Wolfe†, W. Yang¶,
M. Young¶, and L. Zaslavsky*

*NASA/Global Modeling and Assimilation Office, Greenbelt, MD
†NSF/National Center for Atmospheric Research, Boulder, CO
‡NOAA/Geophysical Fluid Dynamics Laboratory, Princeton, NJ
§Massachusets Institute of Technology, Cambridge, MA
¶NOAA/National Centers for Environmental Prediction
‖DoE/Argonne National Laboratory, Argonne, IL
**DoE/Los Alamos National Laboratory, Los Alamos, NM
††University of Michigan, Ann Arbor, MI

ESMF Joint Specification Team
Email: esmf_tech@ucar.edu

***Abstract*— The Earth System Modeling Framework (ESMF) project, which consists of Earth scientists and computational experts from major U.S. Earth modeling centers, is developing a robust, flexible set of software tools to enhance ease of use, performance portability, interoperability, and reuse in climate, numerical weather prediction, and data assimilation applications. The ESMF allows diverse scientific groups to leverage common software to solve routine computational problems such as efficient data communication, model component coupling and sequencing, time management, and parameter specification. In an open dialogue with the broader community, this effort is developing a software interface specification so that groups working at different institutions and in different disciplines can generate interoperable software components.**

I. Introduction

The scientific challenge of developing advanced Earth system applications is a daunting task. Independently developed components may have incompatible interfaces or may be written in different computer languages. The high-performance computer (HPC) platforms required by numerically intensive Earth system applications are complex, varied, rapidly evolving and multi-part systems themselves. Since the market for high-end platforms is relatively small, there is little robust middleware available to buffer the modeler from the difficulties of HPC programming. To complicate matters further, the collaborations required to develop large Earth system applications often span initiatives, institutions and agencies, involve geoscience, software engineering, and computer science communities, and cross national borders.

Corresponding author address: Dr. Arlindo da Silva, Global Modeling and Assimilation Office, NASA/Goddard Space Flight Center, Code 900.3, Greenbelt, MD 20771. Email: Arlindo.daSilva@nasa.gov

The Earth System Modeling Framework (ESMF) project is a concerted response to these challenges. Its goal is to increase software reuse, interoperability, ease of use and performance in Earth system models through the use of a common software framework, developed in an open manner by leaders in the modeling community. The ESMF addresses the technical and to some extent the cultural aspects of Earth system modeling, laying the groundwork for addressing the more difficult scientific aspects, such as the physical compatibility of components, in the future.

The ESMF is a three-year project funded by the NASA Earth Science Technology Office. It is structured as three interlinked projects: the first focuses on development of the core ESMF framework; the second, deployment of the framework in climate and numerical weather prediction models; and the third, deployment of the framework in a number of data assimilation systems. Collaborators on the three proposals include the NSF/National Center for Atmospheric Research, NASA/Goddard Space Flight Center, the Massachusetts Institute of Technology, the University of Michigan, DOE/Argonne National Laboratory, DOE/Los Alamos National Laboratory, the NOAA/Geophysical Fluid Dynamics Laboratory, and the NOAA/National Centers for Environmental Prediction.

This paper is organized as follows. The rationale and objectives of the ESMF are presented in section II, while section III delineates the technical and scientific scope of the ESMF. An overview of the ESMF architecture appears in section IV, with a discussion about the relation of the ESMF to other scientific software frameworks included in section V. Concluding remarks, current development status and future plans for the ESMF are in section VI.

II. Rationale and objectives of the ESMF

The value of interoperable codes for Earth system modeling and data assimilation has become increasingly apparent. Forecast requirements for both weather and climate are becoming more stringent, and data assimilation is becoming a crucial component for both prediction and analysis of the climate record. Despite the advantages of code interoperability, very little progress has been made toward this goal. Groups may share low-level parameterizations but fail to share more high level components such as a full ocean model. In a coupled ocean-atmosphere model, for example, the task of replacing one ocean model with another model from a different organization often requires a major redevelopment effort. Typically, the imported ocean model undergoes substantial interface modifications, becoming in the process a variant of the original model, which is no longer maintained by the developing organization. One of the main goals of the ESMF is to develop a standard interface wich will clearly separate model component and couplers, so that interoperable components can be shared and reused.

The difficulty of the climate/weather prediction problem increases with the complexity of the model and the number and different types of satellite observations to be assimilated. The more complex the systems the harder it will be to require interoperability of components. At the same time, it is precisely for the more complex systems — such as coupled climate models and data assimilation systems — that interoperability is most important. To further complicate the task before us, the requirements for much more sophisticated modeling and assimilation methods have coincided with major changes in computer architecture. The need to utilize distributed memory architectures, or even more daunting, to mix in the same computer and the same application distributed and shared memory programming models, has placed a huge burden on the already strained software development efforts of most groups doing earth system modeling. It has also led to the realization that the old approach, in which each center develops its own solutions, is not just preventing the interchange of scientific codes, but is simply becoming unaffordable. A common *reusable and interoperable* solution to these problems must be developed. This is the rationale for the ESMF.

A particular challenge to the ESMF is that applications in the operational environment with its strict requirements on turnaround cannot afford a marked performance degradation merely to support interoperability. On the positive side, the distinct benefits to be gained from published standards in coding and common interfaces to access data streams should be a strong enabling factor in the transition of new developments from the external community to the operational groups. With these factors in mind, the specific objectives of the ESMF are to provide the following benefits:

a) Facilitate the exchange of scientific codes (interoperability) so that researchers may more easily take advantage of the wealth of resources that are available in the US in smaller-scale, process modeling and to more easily share experience among diverse large-scale modeling and data assimilation efforts.

b) Promote the reuse of standard, non-scientific software, the development of which now accounts for a substantial fraction of the software development budgets of large groups. Any center developing or maintaining a large system for NWP, climate or seasonal prediction, data assimilation, or basic research will have to solve very similar software engineering and routine computational problems.

c) Focus community resources to deal with architectural changes and the lack of quality commodity middleware. The non-scientific parts of the codes that would be dealt with in a common framework are also the most sensitive to architectural changes and *middleware* quality.

d) Present the computer industry with a unified, well defined and well documented task for them to address in their software design. The scientific community's influence with the industry is much diminished, but it will be even smaller if it is exercised separately by five or six centers.

e) Share the overhead costs of the housekeeping aspects of software development: documentation and configuration management. These are the efforts that are most easily neglected when corners have to be cut.

f) Provide institutional continuity to model and data assimilation development efforts. Most US modeling and data assimilation efforts are necessarily tied to only a few individuals, and centers are hard-pressed to maintain continuity that transcends them. The competitive job market we are in will result in shorter tenure for programmers. At the same time, the increasing complexity of both our systems and the technology will produce more reliance on software specialists — and less on scientists — to maintain these aspects of the systems. Both of these factors will contribute to a more unstable workforce and much greater difficulty in maintaining *institutional continuity*. A framework can help us do this by having a much larger institution to support it — the whole community. Also by having codes depend on a common, well-known framework, it will be easier to find and train new people to continue a line of development.

III. Scope of the ESMF

The specific focus of the ESMF is Earth system modeling, including ocean, atmosphere, land and sea-ice models. Targeting this diverse, but bounded domain permits a design and implementation strategy that satisfies user requirements to an extent not possible with generic software. There are 15 initial testbed codes for the ESMF [1]. These represent major national modeling and data assimilation efforts such as the NCEP Global Forecast System [6], GFDL Flexible Modeling System (FMS) [2], the NCAR Community Climate System Model (CCSM) [3], and newly formed NASA's Global Modeling and Assimilation Office[1] (GMAO). These efforts are drawn from both research and operational applications in climate, weather

[1]Formely NASA's Seasonal to Interannual Prediction Project (NSIPP) and Data Assimilation Office (DAO).

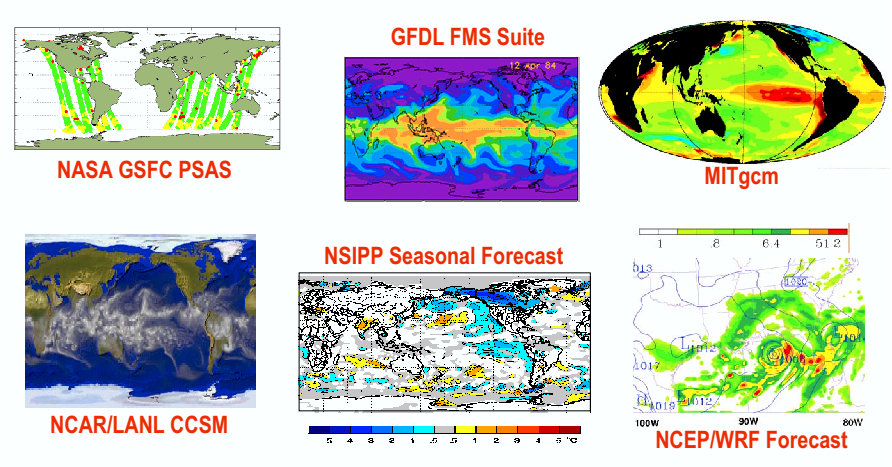

Fig. 1. Sample models to be used in ESMF interoperability demonstrations. At present these modeling systems are all independent from one another and computationally incompatible. Under ESMF these systems, together with several other modeling efforts, will continue to evolve and develop independently but the resulting software need no longer be computationally incompatible.

and data assimilation. The testbed modeling systems span a variety of discrete griding approaches, numerical time-stepping techniques, software programming paradigms and hardware platforms. However, the overall domain is sufficiently focused to allow the identification of common domain specific data and control constructs and abstractions.

The ESMF project aims to impact Earth system modeling on a wide range of time scales. In addition to the adoption of the ESMF in the 15 testbed codes, we are planning a series of 8 interoperability demonstrations involving models and data assimilation systems that have never been coupled before. Some of the problems addressed in these interoperability experiments are:

Numerical Weather Prediction.

In the field of numerical weather prediction large-scale (global and regional) atmospheric simulation has been a key ingredient of forecasting since the 1950's. Forecast skill is sensitive to boundary conditions. As part of the ESMF project, operational forecast model configurations involving coupling an atmospheric simulation to an interactive ocean at the lower boundary will be demonstrated. In figure 1 this corresponds to connecting the NCEP/WRF forecast models with the GFDL FMS Suite. This sort of capability has direct relevance to more accurate forecasting of tropical storm tracks, which is of significance to coastal communities and maritime enterprises.

Data Assimilation.

Although data assimilation is recognized as an important tool for validation and quantitative model development, the complexity of developing state-of-the-art assimilation algorithms have prevented some of the major climate modeling centers from developing such capability. By including data assimilation systems from NASA GMAO and NCEP, one the interoperability experiments represented in Fig. 1 will bring for the first time atmospheric data assimilation capabilities to the NSIPP model.

Seasonal and interannual forecasts.

Longer time-scale forecasts of seasonal to interannual (SI) phenomena are an increasing area of interest with the possible potential to anticipate persistent regional weather shifts such as increased likelihood of drought or flooding well in advance. Evidence suggests that accurate forecasting of ocean conditions, for example the El Nino - La Nina phenomena in the Pacific, and land surface conditions, such as soil moisture content, can improve seasonal and interannual predictability. Work on coupling sea-ice into SI prediction systems under ESMF will allow ideas on improving SI predictability to be explored and applied. In figure 1 this corresponds to connecting the NSIPP Seasonal Forecast models with elements of the NCAR/LANL CCSM suite.

Climate Change.

In the area of improved decadal and centennial climate change estimates, interoperability demonstrations that involve new interchanges of climate model components will be configured using ESMF. These configurations will lay the foundation for studies that can examine the impact of large component interchange on climate simulation trajectories. In figure 1 this corresponds to connecting MITgcm model configurations with elements of the NCAR/LANL CCSM suite and with elements of the GFDL FMS suite.

In each of these cases ESMF provides a universal software framework that simplifies and streamlines the technical steps involved in constructing and executing high-performance, multi-component Earth science applications. The role of ESMF is to facilitate interoperation between the components, impacting productivity by allowing science and engineering users of Earth system models to focus on modeling tasks.

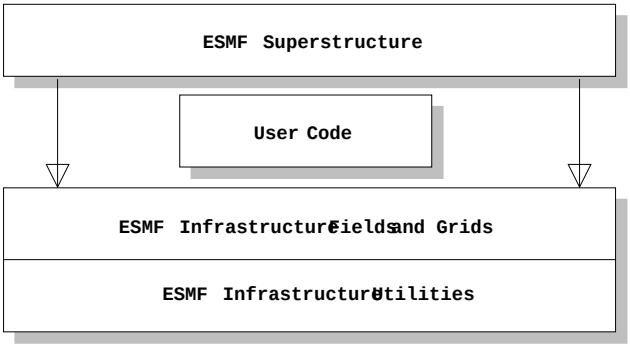

Fig. 2. Schematic of ESMF "sandwich" architecture. In this design the framework consists of two parts. An upper level **Superstructure** layer and a lower-level **Infrastructure** layer. User code is sandwiched between these two layers.

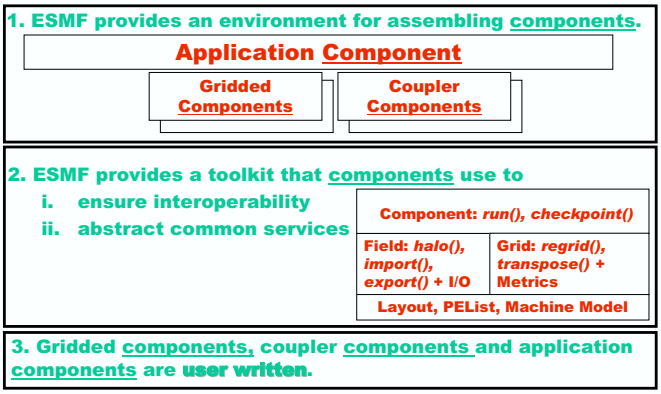

Fig. 3. The ESMF programming paradigm defines how an overall application is constructed. An application is an assembly of one or more gridded and coupler components (1). Components may make use of the ESMF **Infrastructure** toolkit (2). All components, gridded components, coupler components and the top-level application component are primarily user written (3).

IV. Architectural Overview

The ESMF architecture is characterized by the layering strategy shown in figure 2. In this architectural pattern user code components that implement the *science* elements of an algorithm, for example code implementing a finite-difference ocean model, are sandwiched between two layers. The upper layer is denoted the **Superstructure** layer and the lower layer the **Infrastructure** layer. The role of the **Superstructure** layer is to provide a shell which encompasses user code and provides a context for interconnecting input and output data streams between components. The key elements of the **Superstructure** layer are described in section IV-B. These elements include the extensible classes that represent envelope user code components, ensuring that all components present consistent interfaces. The **Infrastructure** layer provides a foundation that component developers can use to build their models. The elements of the **Infrastructure** layer include constructs to support parallel processing with data types tailored to Earth science applications, specialized libraries to support consistent time and calendar management, in addition to performance, error handling and scalable I/O tools. The **Infrastructure** layer is described in section IV-C. A hierarchical combination of **Superstructure**, user code components and Infrastructure are joined together to form what is termed an *application component* in the ESMF programming paradigm.

A. Programming Paradigm

A complete, executable assembly of **Superstructure**, user code components and **Infrastructure** collectively forms an ESMF *application component*. Figure 3 shows the generic structure of an ESMF application component. This figure shows a single tier composition involving three components. It captures the essence of the most basic composition paradigm that ESMF employs, although multi-tier composition is also supported in which components are recursively nested. An application is composed by connecting together one or more numerical simulation or other user code components within an overall ESMF based environment (Fig. 3, top panel.) User provided components are written or modified to fit within the ESMF environment, with interfaces consistent with ESMF's high-level **Superstructure**. A foundation-level **Infrastructure** is also provided, Fig. 3 (middle panel), to both accelerate user code development and ensure compatibility and consistency between components across diverse hardware platforms.

B. Superstructure

The ESMF **Superstructure** layers in an application furnish a unifying context within which user components are interconnected. For example an atmospheric model may use a particular land-surface model in calculating simulated evaporative fluxes. The flow of data and sequence of computation between atmospheric model term evaluations and land-surface model term evaluations would be prescribed in the **Superstructure** layer. Under ESMF user code components are constructed or adapted to fit within this **Superstructure** layer. This ensures that large components can be interchanged. There may still be issues of physical consistency between components, but ensuring that all components comply with the requirement to fit within an ESMF **Superstructure** layer eliminates computational incompatibilities.

The **Superstructure** layer provides the foundation for a flexible mechanism to address physical consistency between components that may use different dimensions or units to represent the same quantity or that may partition physical data differently. Classes called *Gridded Components*, *Coupler Components*, *Import States* and *Export States* are used within the **Superstructure** layer to achieve this flexibility. We define these classes below:

1) Import and Export State Classes: User code components under ESMF use special interface objects for component to component data exchanges. These objects are of type Import State and Export State. These special types support a variety of methods that allow user code components to, for example,

fill an export state object with data to be shared with other components or query an import state object to determine its contents. In keeping with the overall requirements for high-performance it is permitted for Import State and Export State contents to use references or pointers to component data, so that costly data copies of potentially very large data structures can be avoided whenever possible. The content of an Import State and an Export State is self-describing.

2) Interface Standards: The Import State and Export State abstractions are designed to be flexible enough, and the ESMF superstructure layer does not mandate a standard for the contents of these states. For example, ESMF does not prescribe the units of quantities exported or imported, instead it provides mechanics to describe the units, memory layout, grid coordinates of the fields in Import States and Export States. This allows the ESMF software to support a range of different policies for physical fields, leaving the establishment of these standards to the specific scientific community for that particular discipline. The interoperability experiments used to demonstrate ESMF make use of the emerging CF standards [18] for describing physical fields. This is a policy choice for that set of experiments. The ESMF software itself can support arbitrary conventions for labeling and characterizing the content of Import and Export States.

3) Gridded Component Class: The Gridded Component class is used for a user component that takes in one Import State and produces one Export State, both based on the same discrete grid familiar to the component. Examples of Gridded Components are major Earth system model components such as land-surface models, ocean models, atmospheric models and sea-ice models. Components used for linear algebra manipulations in a state-estimation or data-assimilation optimization procedure are also created as Gridded Components.

4) Coupler Component Class: The other top-level component class supported in the current ESMF architecture is a Coupler Component class. This class is used for components that take one or more Import States as input and map them through spatial and temporal interpolation or extrapolation onto an output Export State. In a Coupler Component it is often the case that the output Export State is on a different discrete grid to that of the Import State(s). The role of Coupler Components is generally mapping the Export States from one or more Gridded Components to the Import State of another Gridded Component. For example, in a coupled ocean-atmosphere simulation a Couple Component would be used to map a set of sea-surface fields in an ocean model to appropriate planetary boundary layer fields in an atmospheric model.

5) Flexible data and control flow: Import States, Export States, Gridded Components and Coupler Components can be arrayed flexibly within a **Superstructure** layer. Using these constructs it is possible to configure a set of concurrently executing Gridded Components joined through a single Coupler Component, the so-called *hub-and-spoke* style shown in Fig. 4. Is is also possible to configure a set of sequentially executing components with multiple pair-wise couplers, using the *Tinker-toy* style depicted in Fig. 5.

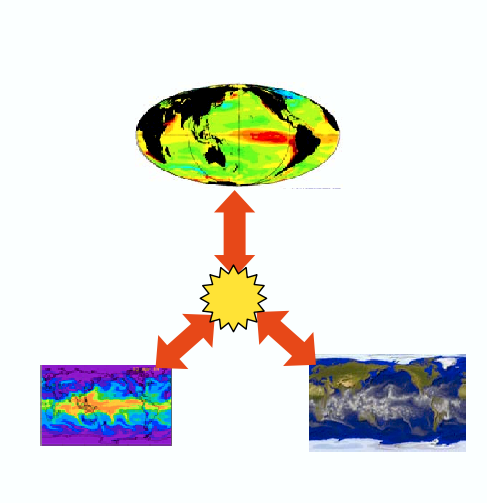

Fig. 4. ESMF can support configurations with a single central Coupler Component. In this case inputs from all Gridded Components are transferred and regridded between all components in one place. The block arrows show how the Coupler Component (symbolized by the star icon) must take inputs from all Gridded Components (symbolized by the model output images) and return data to all Gridded Components.

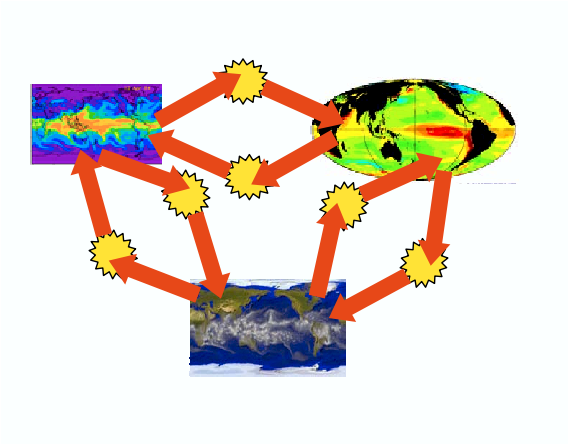

Fig. 5. ESMF also supports configurations with multiple point to point Coupler Components. These take inputs from one Gridded Component and transfer and regrid the data for passing to another Gridded Component. The block arrows show the flow of data between point to point pairings of Coupler Components (symbolized by the star icons) and Gridded Components (symbolized by the model output images).

C. Infrastructure

Figure 6 illustrates three Gridded Components each defined on its own grid. The associated Coupler components are required to interpolate fields from one grid to another, and account for the different units, memory layout and domain decompositions. A common *clock* is also required to handle those operations involving time. The **Infrastructure** layer contains a set of classes that address these issues and assist in managing overall system complexity. We describe these classes below: between the different grids,

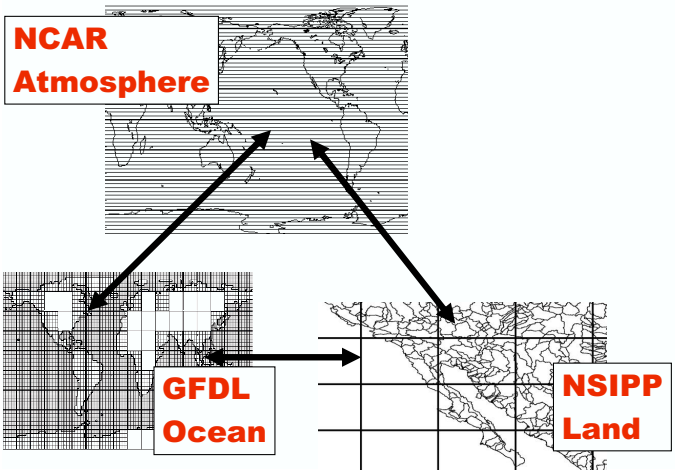

Fig. 6. Schematic showing the coupling of components that use different discrete grids and different time-stepping. In this example component *NCAR Atmosphere* might use a spectral grid based on spherical harmonics, component *GFDL Ocean* might use a latitude-longitude grid but with a patched decomposition that does not include land masses and component *NSIPP Land* might use a mosaic based grid for representing vegetation patchiness and a catchment area based grid for river routings. The **Infrastructure** layer contains tools to help develop software for coupling between components on different grids, mapping between components with different distributions in a multi-processor compute environment and to synchronize events between components with different time-stepping intervals and algorithms.

1) Field and Array Classes: The *Field* and *Array* classes contain data and descriptive physical and computational attribute information. The physical attributes include information that describes the units of the data. The computational attributes include information on the layout in memory of the field data.

2) Physical Grid Class: The *Physical Grid* class is an extensible class that holds discrete grid information. It has subtypes that allow it to serve as a container for the full range of different physical grids that might arise in a coupled system. In the example in figure 6 objects of type Physical Grid would hold grid information for each of the spectral grid, the latitude-longitude grid, the mosaic grid and the catchment grid.

3) Regrid Class: The *Regrid* class is an extensible class that allows remapping between a field on one physical grid to a field on a different physical grid [19]. It supports precomputation of grid interpolation weights and allows user selectable corrections for global or local conservation requirements. Regrid is designed to be scalable on parallel platforms. When mapping between grids the Regrid class utilizes the Physical Grid object and Field and Array objects.

4) Distributed Grid Class: The *Distributed Grid* class is used to represent the decomposition of a data structure into sub-domains, typically for parallel processing purposes. The class is designed to support a generalized "ghosting" for tiled decompositions of finite difference, finite volume and finite element codes.

5) Time and Calendar Management Class: To support synchronization between components *Time* and *Calendar* classes along with an associated *Clock* class are provided. These classes allow Gridded and Coupler Component processing to be latched to a common controlling clock.

6) I/O Classes: The **Infrastructure** layer defines a set of *I/O* classes for storing and retrieving Field and Grid information to and from persistent storage. The I/O classes support a range of standard formats including binary I/O and netCDF, HDF5 and GRIB based I/O.

7) Communication Class: To provide a mechanism for ensuring performance portability ESMF defines a *Communication* class. This class provides a set of high-level platform independent interfaces to performance critical parallel processing communication routines. These routines can be tuned to specific platforms to ensure optimal parallel performance on many platforms. The Communication class includes reduction operations, transpose or redistribution operations and halo or ghost operations.

8) Logging and Profiling Class: The *Logging* and *Profiling* classes are designed to aid in managing the complexity of multi-component applications. They provide ESMF with a unified mechanism for notification messages, for timing and counting events.

9) Configuration Attribute Class: . The *Configuration Attribute Class* provides models with means to maintain ad-hoc sets of parameters. These parameters can be related to physical terms (for example mixing coefficients, length scales or time scales) or can be related to computational aspects (for example directory names, output and input locations). The configuration attributes element of ESMF will enable these parameters or attributes to be recorded and provided services for setting attributes from human readable text files and for saving attributes to persistent storage in appropriate formats.

V. Relation to Other Software Frameworks

In the Earth science community, there have been several efforts to manage software complexity through the development of modeling frameworks. In the US, initial domain specific framework efforts include the GFDL Flexible Modeling System [2], the Goddard Earth Modeling System [7], the Weather Research and Forecast Model [8], the Wrapper toolkit [9], the Community Climate System Model [3], the Pilgrim communications toolkit [10] and the Model Coupling Toolkit [11]. The ESMF software framework architecture aims to unify, standardize and extend these efforts.

In Europe, the *Program for Integrated Earth System Modeling* (PRISM) [12] is funded by the European Comission for defining a standard set of interfaces to an unified coupling mechanism which manages the data exchange and synchronization of each component. Specifically, PRISM provides an infrastructure to easily a) assemble earth system model components, b) to launch/monitor complex earth system models, and c) to access, analyze and share results accross the wide community. PRISM and ESMF are complementary approaches to modeling complexity problem. PRISM focus on the runtime environment and the coupling infrastructure while the ESMF focus on the modeling component code infrastructure

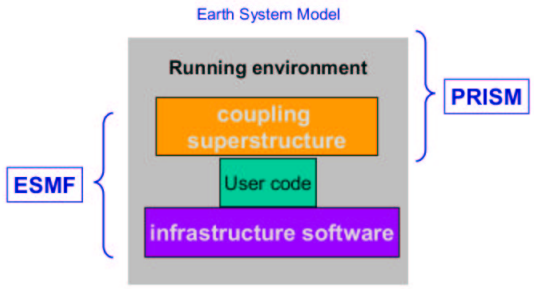

Fig. 7. The relative scopes of the ESMF and PRISM development efforts.

and the modeling coupling superstructure; these complementary scopes are illustrated in Fig. 7. These two projects have established a working relationship to ensure the compatibility of key standards and interfaces.

In the science disciplines of computational chemistry and general relativity the NWChem [4] and Cactus [5] frameworks respectively are examples of established multi-institutional HPC framework efforts. Both systems provide high-level abstractions and programming environments framed in terms of the science workflow of the domain they service, accelerating the development and construction of high-performance simulation tools. Like ESMF, both systems emphasize a standard software platform for modular science component development.

The focused domain approach being taken by ESMF leverages on lessons and technology from more general high-performance computing frameworks such as POOMA [13], Overture [14] and VSIPL [15]. These efforts support more general abstraction and do not focus on a specific discipline such as Earth science. However, these general purpose framework efforts do provide many useful abstractions for key mathematical constructs on high-performance parallel systems. The ESMF architecture draw on these ideas internally.

An important paradigm that many software frameworks employ is component based development. Ideas and technology from mainstream component based programming environments such as CORBA [16] and from the CCA [17] high-performance oriented component programming environment are also being incorporated into the ESMF architecture. However, the component model supported by the ESMF architecture is tailored to the requirements of high-performance Earth science models.

VI. Concluding Remarks

The Earth System Modeling Framework (ESMF) collaboration is building high-performance, flexible software infrastructure to increase ease of use, performance portability, interoperability, and reuse in climate, numerical weather prediction, data assimilation, and other Earth science applications. We are aiming to create a framework usable by individual researchers as well as major operational and research centers, and seek to engage the community in its development.

The ESMF project is about half way in its 3 year funding cycle. The second ESMF Community Meeting was held on May 15, 2003 at the Geophysical Fluid Dynamics Laboratory in Princeton, N.J. During this meeting, we presented an initial (Version 1) ESMF Application Programming Interface (API) and prototype code. The very first demonstration interoperability experiments involving 2 operational data assimilation systems from NOAA's NCEP and NASA's GMAO, as well as climate model components from NCAR, GFDL and GMAO are scheduled to be completed in the Fall of 2003. Version 2 of the ESMF API is expected to be delivered in the Spring 2004, with the full conversion of the 15 testbed codes and remaining interoperability experiments to be completed in the Fall of 2004.

For additional information on the ESMF project, as well as extensive documentation and source code please consult the project website [21].

Acknowledgment

The ESMF project is funded by the NASA Earth Science Technology Office (ESTO) Computational Technologies (CT) Project under the Cooperative Agreement Notice (CAN) entitled: *Increasing Interoperability and Performance of Grand Challenge Applications in the Earth, Space, Life and Microgravity Sciences.*

References

[1] The ESMF Joint Milestone Code set. http://www.esmf.ucar.edu/main_site/esmf_apps.html

[2] The Flexible Modeling System. *Geophysical Fluid Dynamics Laboratory*. http://www.gfdl.gov/fms

[3] The NCAR Climate System Model, Version One. *Journal of Climate* 1998, 11(1115-1130). *B.A. Boville and P.R. Gent*

[4] NWChem, A Computational Chemistry Package for Parallel Computers. *Pacific Northwest National Laboratory*, 2002, Richland, Washington.

[5] The Cactus Framework and Toolkit: Design and Applications. *Vector and Parallel Processing - VECPAR'2002 T. Goodale, G. Allen, G. Lanferma, J. Masso. T. Radko, E. Seidel and J. Shalf*

[6] NCEP's Global Forecasting System (GFS), see `http://www.emc.ncep.noaa.gov/`

[7] The GEOS-3 Data Assimilation System. *DAO Office Note 97-06* 1997, NASA Goddard Space Flight Center. *Office Note Series on Global Modeling and Data Assimilation*

[8] Development of a Next Generation Regional Weather Research and Forecast Model. *Proceedings of Ninth ECMWF Workshop on the Use of High Performance Computing in Meteorology* 2001, World Scientific Press. *J.G. Michalakes, S. Chen, J. Dudhia, L. Hart, J. Klemp, J. Middlecoff and W. Skamarock.*

[9] A strategy for tera-scale climate modeling. *Proceedings of Ninth ECMWF Workshop on the Use of High Performance Computing in Meteorology* 2001, World Scientific Press. *C. Hill, A. Adcroft, D. Jamous and J. Marshall*

[10] Parallel Grid Manipulation in Earth Science Calculations *Proceedings of 3rd International Meeting on Vector and Parallel Processing* 1999, Springer-Verlag Lecture Notes in Computer Science. Volume 157(666-679). *W. Sawyer, L. Takacs, A. da Silva and P.M. Lyster*

[11] The Model Coupling Toolkit. *Proceedings of the International Conference on Computer Science*, 2001, Springer-Verlag Lecture Notes on Computer Science Volume 2073(185-194). *J. Larson, R. Jacob, I. Foster and J. Guo*

[12] PRISm System Specification handbook - Version 1, 2003. *PRISM report Series* No. 1, ISBN 90-369-2217-8. *E. Guilyardi, R. Budich, . Brasseus, G. Komen.*.
[13] The POOMA Framework. *Computers in Physics* 1998, vol. 12, number 5(453-459). *J.Reynders and P.Dubois*
[14] Overture: An Object-Oriented Framework for Solving Partial Differential Equations on Overlapping Grids *SIAM Conference on Object Oriented Methods for Scientific Computing*, 199. *D.L. Brown, W.D. Henshaw and D.J. Quinlan*
[15] A portable object-based parallel library and framework for real-time radar signal processing. *Scientific Computing in Object-Oriented Parallel Environments* 1997, Springer-Verlag, Berlin Germany. *C. DeLuca, C. Heisey, R. Bond and J.M. Daly*
[16] The Common Object Request Broker: Architecture and Specification. *OMG Document* 1995.
[17] Toward a Common Component Architecture for High Performance Scientific Computing. *Proceedings of the Conference on High Performance Distributed Computing*, 1999. *R. Armstrong, D. Gannon, A. Geist, K. Keahey, S. Kohn, L. McInnes, S. Parker and B. Smolinski*
[18] NetCDF Climate and Forecast Metadata Conventions. http://www.cgd.ucar.edu/cms/eaton/cf-metadat
[19] First- and Second-Order Conservative Remapping Schemes for Grids in Spherical Coordinates. *Monthly Weather Review* 1999, 127(2204-2210). *P.W. Jones* http://www.acl.lanl.gov/climate/software/SCRIP
[20] The Second ESMF Community Meeting *Geophysical Fluid Dynamics Laboratory* http://www.gfdl.gov/esmf
[21] The ESMF Project Web Site. http://www.esmf.ucar.edu.